GEETHANJALI INSTITUTE OF SCIENCE &TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapuramu)

3rd Mile, Bombay Highway, Gangavaram (V), Kovur(M), SPSR Nellore (Dt),
Andhra Pradesh, India- 524137

C - Programming & Data Structures

L A B O R A T O R Y O B S E R V A T I O N

Department of

COMPUTER SCIENCE &ENGINEERING

Name	
Roll No.	
Class	I B.TECH. I SEM
Branch	CSE
Regulation	R20
Name of the Lab	C – PROGRAMMING & DATA STRUCTURES
Academic year	2020-21

VISION (GIST)

- To emerge as a leading Engineering institution imparting quality education

MISSION (GIST)

- IM_1 Implement Effective teaching-learning strategies for quality education
- IM_2 Build Congenial academic ambience for progressive learning
- IM_3 Facilitate Skill development through Industry-Institute initiatives
- IM_4 Groom environmentally conscious and socially responsible technocrats

DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING

VISION

- To develop as a lead learning resource centre producing skilled professionals

MISSION

- DM_1 Provide dynamic and application oriented education through advanced teaching learning methodologies
- DM_2 Create sufficient physical infrastructural facilities to enhance learning
- DM_3 Strengthen the professional skills through effective Industry- Institute Interaction
- DM_4 Organize personality development activities to inculcate life skills and ethical values

GEETHANJALI INSTITUTE OF SCIENCE &TECHNOLOGY *(Approved*

by AICTE, New Delhi & Affiliated to JNTUA, Anantapuramu)

3rd Mile, Bombay Highway, Gangavaram (V), Kovur(M), SPSR Nellore (Dt),
Andhra Pradesh, India- 524137

C - Programming & Data Structures

L A B O R A T O R Y O B S E R V A T I O N

Department of

COMPUTER SCIENCE &ENGINEERING

Name	
Roll No.	
Class	I B.TECH. I SEM
Branch	CSE
Regulation	R20
Name of the Lab	C – PROGRAMMING & DATA STRUCTURES
Academic year	2020-21

C PROGRAMMING AND DATA STRUCTURES LAB SYLLABUS (20A05201P)

COURSE OUTCOMES

On successful completion of this course, the student will be able to

- Demonstrate the basic concepts of C programming language.
- Develop C programs using functions, arrays, strings, structures and pointers.
- Illustrate the concepts of Stacks and Queues and apply them.
- Design operations on Linked lists.
- Analyze the concept of trees and apply various Binary tree traversal techniques.
- Develop searching and sorting methods.

List of Experiments

Week l

Write C programs that use both recursive and non-recursive functions

i) To find the factorial of a given integer.
ii) To find the GCD (greatest common divisor) of two given integers.
iii) To solve Towers of Hanoi problem.

Week 2

a) Write a C program to find both the largest and smallest number in a list of integers.

b) Write a C program that uses functions to perform the following:

i) Addition of Two Matrices

ii) Multiplication of Two Matrices

Week 3

a) Write a C program that uses functions to perform the following operations:

i) To insert a sub-string in to a given main string from a given position.

ii) To delete n characters from a given position in a given string.

Week 4

a) Write a C program that displays the position or index in the string S where the string T begins, or – 1 if S doesn‘t contain T.

b) Write a C program to count the lines, words and characters in a given text.

Week 5

a) Write a C Program to perform various arithmetic operations on pointer variables.

b) Write a C Program to demonstrate the following parameter passing mechanisms:

i) call-by-value ii) call-by-reference

Week 6

Write a C program that uses functions to perform the following operations:

i) Reading a complex number
ii) Writing a complex number
iii) Addition of two complex numbers
iv) Multiplication of two complex numbers (Note: represent complex number using a structure.)

Week 7

Write C programs that implement stack (its operations) using

i)Arrays ii) Pointers

Week 8

Write C programs that implement Queue (its operations) using

i)Arrays ii) Pointers

Week 9

Write a C program that uses Stack operations to perform the following:

i)Converting infix expression into postfix expression

ii) Evaluating the postfix expression

Week 10

Write a C program that uses functions to perform the following operations on singly linked list.

i)Creation ii) Insertion iii) Deletion iv) Traversal

Week 11

Write a C program that uses functions to perform the following operations on Doublylinkedlist.

i)Creation ii) Insertion iii) Deletion iv) Traversal

Week 12

Write a C program that uses functions to perform the following operations on circular linkedlist.

i) Creation ii) Insertion iii) Deletion iv) Traversal

Week 13 Write a C program that uses functions to perform the following:

i) Creating a Binary Tree of integers

ii) Traversing the above binary tree in preorder, inorder and postorder.

Week 14

Write C programs that use both recursive and non-recursive functions to perform the following searching operations for a key value in a given list of integers:

i) Linear search ii) Binary search

Week 15

Write a C program that implements the following sorting methods to sort a given list of integers in ascending order i) Bubble sort ii) Selection sort iii) Insertion sort

INDEX

Name of the Student :	*Roll No:*
*Class&Branch:***I**B.Tech. **I** - sem, CSE	*Lab :* ***CP*DS**

S.No.	Name of the experiment	Page No.	Date	Signature
1	Factorial of a given integer without recursion.			
2	Factorial of a given integer using recursion.			
3	GCD of two given integers without recursion.			
4	GCD of two given integers using recursion			
5	Towers of Hanoi without recursion			
6	Towers of Hanoi using recursion			
7	Largest and smallest number in a list of integers.			
8	Functions to perform the following operations: Addition of two matrices Multiplication of two matrices			
9	Functions to perform the following operations: i. To insert a substring into a given main string from a given position. ii. To delete n characters from a given position in a given string.			
10	Position or index in the string S where the string T begins, or – 1 if S doesn't contain T.			
11	Count the lines, words and characters in a given text.			
12	Perform various arithmetic operations on pointer variables			
13	Demonstrate the following parameter passing mechanisms: i) call-by-value ii) call-by-reference			

S.No.	Name of the experiment	Page No.	Date	Signature
14	Functions to perform the following operations: i) Reading a complex number ii) Writing a complex number iii) Addition of two complex numbers iv) Multiplication of two complex numbers			
15	Stack (its operations) using an array.			
16	Stack (its operations) using pointers.			
17	Queue (its operations) using an array			
18	Queue (its operations) using pointers.			
19	Stack operations for converting an Infix expression to postfix expression			
20	Stack operations for evaluating postfix expression			
21	functions to perform the following operations on singly linked list. i) Creation ii) Insertion iii) Deletion iv) Traversal			
22	functions to perform the following operations on Doubly linked list. i) Creation ii) Insertion iii) Deletion iv) Traversal			
23	functions to perform the following operations on circular linked list. i) Creation ii) Insertion iii) Deletion iv) Traversal			
	functions to perform the following: i) Creating a Binary Tree of integers ii) Traversing the above binary tree in preorder, inorder and postorder.			
24	recursive and non-recursive functions to perform the following searching operations for a key value in a given list of integers: i) Linear search ii) Binary search			
25	sorting methods to sort a given list of integers in ascending order i) Bubble sort ii) Selection sort iii) Insertion sort			

Signature of the Staff member

Exp.No. 1	C programs that use both recursive and non-recursive functions i) To find the factorial of a given integer. ii) To find the GCD (greatest common divisor) of two given integers. iii) To solve Towers of Hanoi problem
Dt.	

1. a) Write a program to find factorial of a given integer using recursion and non-recursion

Source Code:-

```
/*Factorial using recursion and non-recursion*/
#include <stdio.h>
int main( )
{
int n, a, b;
printf("Enter any number\n");
scanf("%d", &n);
  a = recfactorial(n);
printf("The factorial of a given number using recursion is %d \n", a);
  b = nonrecfactorial(n);
printf("The factorial of a given number using nonrecursion is %d ", b);
}
intrecfactorial(int x)
{
int f;
if(x == 0)
  {
return(1);
  }
else
  {
   f = x * recfactorial(x - 1);
return(f);
  }
}
intnonrecfactorial(int x)
{
inti, f = 1;
for(i = 1;i <= x; i++)
  {
    f = f * i;
  }
return(f);
}
```

Input and Output:-

1. b) Write a program to find GCD of given two integers using recursion and non-recursion.

Source Code:-

```
/* GCD of two integers using recursion and non-recursion*/
#include <stdio.h>
int main( )
{
int a, b, c, d;
printf("Enter two numbers a, b\n");
scanf("%d%d", &a, &b);
  c = recgcd(a, b);
printf("The gcd of two numbers using recursion is %d\n", c);
  d = nonrecgcd(a, b);
printf("The gcd of two numbers using nonrecursion is %d", d);
}
intrecgcd(int x, int y)
{
if(y == 0)
  {
return(x);
  }
else
  {
return(recgcd(y, x % y));
  }
}

intnonrecgcd(int x, int y)
{
int z;
while(x % y != 0)
  {
   z = x % y;
   x = y;
   y = z;
  }
return(y);
}
```

Input and Output:-

1 c) Write a program to solve Towers of Hanoi problem using recursion and non-recursion.

Source Code:-

```
/*Towers of Hanoi using recursion and non-recursion*/
#include<stdio.h>
/* Non-Recursive Function*/

voidhanoiNonRecursion(intnum,charsndl,charindl,chardndl)
{
  char stkn[50],stksndl[50],stkindl[50],stkdndl[50],stkadd[50],temp;
inttop,add;
top=NULL;
one:
if(num==1)
   {
printf("\nMove top disk from needle %c to needle %c ",sndl,dndl);
goto four;
   }
two:
top=top+1;
stkn[top]=num;
stksndl[top]=sndl;
stkindl[top]=indl;
stkdndl[top]=dndl;
stkadd[top]=3;
num=num-1;
sndl=sndl;
temp=indl;
indl=dndl;
dndl=temp;
goto one;
three:
printf("\nMove top disk from needle %c to needle %c ",sndl,dndl);
top=top+1;
stkn[top]=num;
stksndl[top]=sndl;
stkindl[top]=indl;
stkdndl[top]=dndl;
stkadd[top]=5;
num=num-1;
temp=sndl;
sndl=indl;
indl=temp;
dndl=dndl;
goto one;
four:
if(top==NULL)
return;
num=stkn[top];
```

```
sndl=stksndl[top];
indl=stkindl[top];
dndl=stkdndl[top];
add=stkadd[top];
top=top-1;
if(add==3)
goto three;
else if(add==5)
goto four;
}

/* Recursive Function*/
void  hanoiRecursion( intnum,char ndl1, char ndl2, char ndl3)
{
if ( num == 1 ) {
printf( "\nMove top disk from needle %c to needle %c.", ndl1, ndl2 );
return;
    }
hanoiRecursion(num - 1,ndl1, ndl3, ndl2 );
printf( "\nMove top disk from needle %c to needle %c.", ndl1, ndl2 );
hanoiRecursion(num - 1,ndl3, ndl2, ndl1 );
}

int main( )
{
int no;
printf("Enter the no. of disks to be transferred: ");
scanf("%d",&no);
if(no<1)
printf("\nThere's nothing to move.");
else
printf("Non-Recursive");
hanoiNonRecursion(no,'A','B','C');
printf("\nRecursive");
hanoiRecursion(no,'A','B','C');
return 0;
}
```

Input and Output:-

Exp.No. 2 a)	Program to find both the largest and smallest number in a list of integers.
Dt.	

Aim:-Write a C program to find both the largest and smallest number in a list of integers.

Source Code:-

```
#include<stdio.h>
int main()
{
int a[50],i,n,large,small;
printf("Enter size of array:");
scanf("%d",&n);
printf("Enter array elements:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
large=small=a[0];
for(i=1;i<n;i++)
{
if(a[i]>large)
large=a[i];
if(a[i]<small)
small=a[i];
}
printf("The largest element is %d",large);
printf("\nThe smallest element is %d",small);
return 0;
}
```

Input and Output:-

Exp.No. 2 b)	C program that uses functions to perform the following: i) Addition of Two Matrices ii) Multiplication of Two Matrices
Dt.	

Aim:-Write a C program that uses functions to perform the following:
i) Addition of Two Matrices ii) Multiplication of Two Matrices
Source Code:-

```
#include<stdio.h>
void input(int a[8][8],intr,int c);
void output(int a[8][8],intr,int c);
void add(int a[8][8],int b[8][8],int s[8][8],intr,int c);
void multiply(int a[8][8],int b[8][8],int s[8][8],intr,intc,int p);
int main( )
{
        int a[8][8],b[8][8],s[8][8],r1,c1,r2,c2,ch;
        printf("\n Enter rows and columns of first matrix:");
        scanf("%d%d",&r1,&c1);
        printf("\n Enter rows and columns of second matrix:");
        scanf("%d%d",&r2,&c2);
        printf("\n Enter the elements of first matrix:");
        input(a,r1,c1);
        printf("\n Enter the elements of second matrix:");
        input(b,r2,c2);
        printf("\n Elements of first matrix are:");
        output(a,r1,c1);
        printf("\n Elements of second matrix are:");
        output(b,r2,c2);
        if((r1==r2)&&(c1==c2))
        {
                add(a,b,s,r1,c1);
                printf("\n Sum of the two matrices is:");
                output(s,r1,c1);
        }
        else
        {
                printf("\n Matrix addition is not possible");
        }
        if(c1==r2)
        {
                multiply(a,b,s,r1,c2,c1);
                printf("\n Product of the two matrices is:");
                output(s,r1,c2);
        }
        else
        {
                printf("\n Matrix multiplication is not possible");
        }
}
```

```
void input(int a[8][8],intr,int c)
{
inti,j;
for(i=0;i<r;i++)
for(j=0;j<c;j++)
scanf("%d",&a[i][j]);
}

void output(int a[8][8],intr,int c)
{
inti,j;
printf("\n");
for(i=0;i<r;i++)
  {
for(j=0;j<c;j++)
     {
          printf("%4d",a[i][j]);
     }
printf("\n ");
  }
}
void add(int a[8][8],int b[8][8],int s[8][8],intr,int c)
{
inti,j;
for(i=0;i<r;i++)
for(j=0;j<c;j++)
          s[i][j]=a[i][j]+b[i][j];
}

void multiply(int a[8][8],int b[8][8],int s[8][8],intr,intc,int p)
{
inti,j,k;
for(i=0;i<r;i++)
for(j=0;j<c;j++)
   {
          s[i][j]=0;
          for(k=0;k<p;k++)
          s[i][j]=s[i][j]+(a[i][k]*b[k][j]);
   }
}
```

Input-Output:

Exp.No. 3	C program that uses functions to perform the following operations: i)To insert a sub-string in to a given main string from a given position. ii) To delete n characters from a given position in a given string.
Dt.	

Aim:- Write a program i) To insert a sub-string in to a given main string from a given position. ii) To delete n characters from a given position in a given string.

Source Code:-

```
#include<stdio.h>
#include<stdlib.h>
Void insertstring(char [],char [],int);
Void delstring(char [], int,int);
int main( )
{
        charstr[30],substr[15];
        intpos,n;
        printf("1.Insert substring\n");
        printf("Enter the string");
        gets(str);
        printf("Enter the substring to insert");
        gets(substr);
        printf("Enter the position to insert");
        scanf("%d",&pos);
        insertstring(str,substr,pos);
        fflush(stdin);
        printf("2.Delete substring\n");
        printf("Enter the string");
        gets(str);
        printf("Enter the position to delete");
        scanf("%d",&pos);
        printf("Enter number of characters to delete");
        scanf("%d",&n);
        delstring(str,pos,n);

}
Void insertstring(char str[],char substr[],intpos)
{
        inti,j;
        char temp[20];
        for(i=0,j=pos-1;str[j]!='\0';i++,j++)
        {
                temp[i]=str[j];
        }
        temp[i]='\0';
        for(i=0,j=pos-1;substr[i]!='\0';i++,j++)
        {
                str[j]=substr[i];
        }
```

```
        for(i=0;temp[i]!='\0';i++,j++)
        {
                str[j]=temp[i];
        }
        str[j]='\0';
        puts(str);
}

Void delstring(char str[ ],intpos,int n)
{
        inti,j;
        char temp[30];
        for(i=0,j=pos-1+n;str[j]!='\0';i++,j++)
        {
                temp[i]=str[j];
        }
        temp[i]='\0';
        for(i=0,j=pos-1;temp[i]!='\0';i++,j++)
        {
                str[j]=temp[i];
        }
        str[j]='\0';
        puts(str);
}
```

Input and Output:-

Exp.No. 4(a)	program that displays the position or index in the string S where the string T begins, or – 1 if S doesn‘t contain T.
Dt.	

Aim:-Write a program tothat displays the position or index in the string S where the string T begins, or – 1 if S doesn‘t contain T.

Source Code :-

```
#include<stdio.h>
#include<string.h>
int main( )
{
char s[30],t[20];
char *found;
/* Entering the main string */
puts("Enter the first string: ");
gets(s);
/* Entering the string whose position or index to be displayed */
printf("Enter the string to be searched: ");
gets(t);
/*Searching string t in string s */
found=strstr(s,t);
if(found)
printf("Second String is found in the First String at %d position.\n",found-s);
else
printf("-1");
}
```

Input and Output:

Exp.No. 4(b)	program to count the lines, words and characters in a given text.
Dt.	

Aim:-Write a C program to count the lines, words and characters in a given text.

Source Code:

```
#include<stdio.h>
int main( )
{
  // declare variables
Char str[200];
int line, word, ch;

  // initialize count variables with zero
line = word = ch = 0;

  // read multiline string
printf("Enter string terminated with ~ :\n");
scanf("%[^~]", str);

  // check every character
for(inti=0; str[i]!='\0'; i++)
  {
    // if it is new line then
    // one line and one word completed
        if(str[i]=='\n')
        {
                line++;
                word++;
        }

    // else it is a character
        else
        {
      // if character is space or tab
      // then one word is also completed
        if(str[i]==' '||str[i]=='\t')
      {
        word++;
        ch++;
      }

      // it was not '\n', sapace or tab
      // it is a normal character
```

```
else {
          ch++;
       }
     }
   }

  // display count values
printf("\nCharacter counts = %d\n", ch);
printf("Word counts = %d\n", word);
printf("Line counts = %d\n", line);

return 0;
}
```

Input & Output:-

Exp.No. 5	a) Write a C Program to perform various arithmetic operations on pointer variables. b) Write a C Program to demonstrate the following parameter passing mechanisms: i) call-by-value ii) call-by-reference
Dt.	

Aim:-a) Write a C Program to perform various arithmetic operations on pointer variables.

b) Write a C Program to demonstrate the following parameter passing mechanisms: i) call-by-value ii) call-by-reference

Program:-

5(a) Program to perform various arithmetic operations on pointer variables

Source Code:

```
#include <stdio.h>
int main( )
{
int m = 5, n = 10, o = 0;
int *p1;
int *p2;
int *p3;
   p1 = &m;    //printing the address of m
   p2 = &n;    //printing the address of n
printf("p1 = %u\n", p1);
printf("p2 = %u\n", p2);
   o = *p1+*p2;
printf("*p1+*p2 = %d\n", o);//point 1
   p3 = p1-p2;
printf("p1 - p2 = %u\n", p3); //point 2
p1++;
printf("p1++ = %u\n", p1); //point 3
p2--;
printf("p2-- = %u\n", p2); //point 4
return 0;
}
```

Input & Output:-

Program:-

5(b) Program to demonstrate the following parameter passing mechanisms: i) call-by-value ii) call-by-reference.

Source Code:

i)call by value

```
#include<stdio.h>
Void swapnum (int num1,int num2 )
{
int tempnum ;
tempnum = num1 ;
 num1 = num2 ;
 num2 = tempnum ;
printf("\nIn swap( ) num1 value is %d",num1);
printf("\nIn swap( ) num2 value is %d",num2);
}

int main( )
{
int num1 = 35, num2 = 45 ;
printf("Before swapping in main():");
printf("\nnum1 value is %d", num1);
printf("\nnum2 value is %d", num2);

  /*calling swap function*/
swapnum(num1,num2);

printf("\nAfter swapping in main():");
printf("\nnum1 value is %d", num1);
printf("\nnum2 value is %d", num2);
return 0;
}
```

Input & Output:-

ii)call by reference

```
#include<stdio.h>
Void swapnum (intp,int q)
{
Int tempnum ;
tempnum = p;
  p = q ;
  q = tempnum ;
printf("\nIn swap( ) num1 value is %d",p);
printf("\nIn swap( ) num2 value is %d",q);
}
int main( )
{
int num1 = 35, num2 = 45 ;
printf("Before swapping in main( ):");
printf("\nnum1 value is %d", num1);
printf("\nnum2 value is %d", num2);

  /*calling swap function*/
swapnum(num1,num2);
printf("\nAfter swapping in main( ):");
printf("\nnum1 value is %d", num1);
printf("\nnum2 value is %d", num2);
return 0;
}
```

Input & Output:-

Exp.No. 6	Write a C program that uses functions to perform the following operations: i) Reading a complex number ii) Writing a complex number iii) Addition of two complex numbers iv) Multiplication of two complex numbers

Source code:-

```
#include<stdio.h>
struct complex
{
        float real;
        floatimg;
};
void output(struct complex);
struct complex add(structcomplex,struct complex);
struct complex sub(structcomplex,struct complex);
struct complex mul(structcomplex,struct complex);
int main( )
{
struct complex c1,c2,result;
int choice;
printf("\n Enter first complex number");
printf("\n\n Enter real part :");
scanf("%f",&c1.real);
printf("\n\n Enter imaginary part :");
scanf("%f",&c1.img);
printf("\n Enter second complex number");
printf("\n\n Enter real part :");
scanf("%f",&c2.real);
printf("\n\n Enter imaginary part :");
scanf("%f",&c2.img);
result=add(c1,c2);
printf("\n Sum of two complex numbers=");
output(result);
result=sub(c1,c2);
printf("\n Difference of two complex numbers=");
output(result);
result=mul(c1,c2);
printf("\n Product of two complex numbers=");
output(result);
return 1;
 }
```

```
void output(struct complex k)
{
if (k.img>=0)
printf("%2.1f+%2.1fi",k.real,k.img );
else
printf("%2.1f-%2.1fi",k.real,(-k.img) );
}

struct complex add(struct complex x,struct complex y)
{
struct complex t;
t.real=x.real+y.real;
t.img=x.img+y.img;
return(t);
}

struct complex sub(struct complex x,struct complex y)
{
struct complex t;
t.real=x.real-y.real;
t.img=x.img-y.img;
return(t);
}
struct complex mul(struct complex x,struct complex y)
{
struct complex t;
t.real=(x.real*y.real)-(x.img*y.img);
t.img=(x.real*y.img)+(x.img*y.real);
return(t);
}
```

Input & Output:-

Exp.No. 7	Write a C program that uses functions to perform the following operations: i) Reading a complex number ii) Writing a complex number iii) Addition of two complex numbers iv) Multiplication of two complex numbers

(a)Stack using array:-

```
#include<stdio.h>
#include<stdlib.h>
#define MAXSIZE 10
int stack[MAXSIZE], top=-1;
void push( );
void pop( );
void display( );
main( )
{
        int choice;
        charch;
        while(1)
        {
                printf("1.PUSH\n2.POP\n3.DISPLAY\n4.EXIT\n");
                printf("\nEnter your choice");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: push();break;
                        case 2: pop();break;
                        case 3: display();break;
                        case 4: exit(0);
                        default: printf("\nInvalid choice");
                }
        }
}
void push( )
{
        int x;
        if(top==MAXSIZE-1)
        {
                printf("Stack Overflow\n");
                return;
        }
        printf("\nEnter the element to push\n");
        scanf("%d",&x);
        top++;
        stack[top]=x;
}
```

```
void pop( )
{
        int x;
        if(top<0)
        {
                printf("Stack Underflow\n");
                return;
        }
        x=stack[top];
        top--;
        printf("The deleted item is %d",x);
}
void display( )
{
        inti;
        if(top<0)
        {
                printf("Stack is empty\n");
                return;
        }
        for(i=top;i>=0;i--)
        printf("%d\n",stack[i]);
}
```

Input & Output:-

(b)Stack using pointers:-

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
void push( );
void pop( );
void display( );
struct node
{
        int info;
        struct node *prev;
};
typedefstruct node NODE;
NODE *top=NULL,*ptr,*temp;
int main( )
{
        intch;
        while(1)
        {
                printf("1.PUSH\n2.POP\n3.DISPLAY\n4.EXIT");
                printf("\nEnter your choice");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:push();break;
                        case 2:pop();break;
                        case 3:display();break;
                        case 4:exit(0);
                        default:printf("Invalid choice");
                }
        }
return 0;
}
void push( )
{
        ptr=(NODE *)malloc(sizeof(NODE));
        printf("Enter the data");
        scanf("%d",&ptr->info);
        if(top==NULL)
        {
                top=ptr;
                top->prev=NULL;
        }
        else
        {
                ptr->prev=top;
                top=ptr;
        }
}
```

```
void pop( )
{
        if(top==NULL)
        {
                printf("Stack is empty\n");
                return;
        }
        ptr=top;
        printf("The item deleted is %d\n",top->info);
        top=top->prev;
        free(ptr);
}
void display( )
{
        if(top==NULL)
        {
                printf("Stack is empty\n");
                return;
        }
        temp=top;
        while(temp!=NULL)
        {
                printf("%d\n",temp->info);
                temp=temp->prev;
        }
}
```

Input & Output:-

Exp.No. 8	Write C programs that implement Queue (its operations) using i) Arrays ii) Pointers

(a) Queue using array:-

```
#include<stdio.h>
#include<stdlib.h>
#define MAXSIZE 10
int Q[MAXSIZE],front=-1,rear=-1;
void insert( );
voiddelet( );
void display( );
int main( )
{
        intch;
        while(1)
        {
                printf("\n1.INSERT\n2.DELETE\n3.DISPLAY\n4.EXIT\n");
                printf("Enter your choice");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:insert();break;
                        case 2:delet();break;
                        case 3:display();break;
                        case 4:exit(0);
                        default:printf("Invalid choice");
                }
        }
return 0;
}
void insert( )
{
        int x;
        if(rear==MAXSIZE-1)
        {
                printf("Queue overflow\n");
                return;
        }
        if(front==-1)
        {
                front=rear=0;
        }
        else
        {
                rear++;
        }
```

```
        printf("Enter an element to insert");
        scanf("%d",&x);
        Q[rear]=x;
}

Void delet( )
{
        int x;
        if(front==-1)
        {
                printf("Queue is empty\n");
                return;
        }
        x=Q[front];
        if(front==rear)
        {
                front=rear=-1;
        }
        else
        {
                front=front+1;
        }
        printf("The element deleted is %d\n",x);
}

void display( )
{
        inti;
        if(front==-1)
        {
                printf("Queue is empty\n");
                return;
        }
        for(i=front;i<=rear;i++)
        printf("%d\t",Q[i]);
}
```

Input & Output:-

(b) Queue using pointer:-

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
        int info;
        struct node *next;
};
typedefstruct node NODE;
NODE *front=NULL,*rear=NULL,*ptr,*temp;
void insert( );
voiddelet( );
void display( );
int main( )
{       intch;
        while(1)
        {
                printf("\n1.INSERT\n2.DELETE\n3.DISPLAY\n4.EXIT\n");
                printf("Enter your choice");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:insert( );break;
                        case 2:delet( );break;
                        case 3:display( );break;
                        case 4:exit(0);
                        default:printf("Invalid choice");
                }
        }
return 0;
}
void insert( )
{
        ptr=(NODE *)malloc(sizeof(NODE));
        printf("Enter the data to insert");
        scanf("%d",&ptr->info);
        ptr->next=NULL;
        if(front==NULL)
        {
                front=rear=ptr;
        }
        else
        {
                rear->next=ptr;
                rear=ptr;
        }
}
```

```
Void delet( )
{
        if(front==NULL)
        {
                printf("Queue is empty\n");
                return;
        }
        printf("The deleted element is %d\n",front->info);
        temp=front;
        if(front==rear)
        {
                front=rear=NULL;
        }
        else
        {
                front=front->next;
        }
        free(temp);
}
void display( )
{
        if(front==NULL)
        {
                printf("Queue is empty\n");
                return;
        }
        temp=front;
        while(temp!=NULL)
        {
                printf("%d\n",temp->info);
                temp=temp->next;
        }
}
```

Input & Output:-

Exp.No. 9	Write a C program that uses Stack operations to perform the following: i) Converting infix expression into postfix expression ii) Evaluating the postfix expression
Dt.	

9(a) Converting infix expression into postfix expression

Source Code:-

```
#include<stdio.h>
#define MAX 50
char stack[MAX];
int top=-1;
void push(char);
char pop( );
int priority(char);
int main( )
{
char a[MAX],ch;
int i;
printf("enter an infix expression:\t");
gets(a);
printf("\n the postfix expression for the given expression is:\t");
for(i=0;a[i]!='\0';i++)
{
ch=a[i];
if((ch>='a')&&(ch<='z'))
printf("%c",ch);
else if(ch=='(')
push(ch);
else if(ch==')')
{
while((ch=pop( ))!='(')
printf("%c",ch);
}
else
{
while(priority(stack[top])>priority(ch))
printf("%c",pop( ));
push(ch);
}
}/*end for loop*/
while(top>-1)
printf("%c",pop( ));
return 0;
}/*end for main*/
```

```
void push(char ch)
{
if(top==MAX-1)
{
printf("STACK OVERFLOW");
return;
}
else
{
top++;
stack[top]=ch;
}
}

char pop( )
{
int x;
if(top==-1)
{
printf("STACK EMPTY");
}
else
{
x=stack[top];
top--;
}
return x;
}

int priority(char ch)
{
switch(ch)
{
case '^':return 4;
case '*':
case '/':return 3;
case '+':
case '-':return 2;
default:return 0;
}
}
```

Input & Output:-

9(b) Evaluating the postfix expression

Source Code:-

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
#define MAX 50
void push(double);
double pop( );
double oper(char,double,double);
int top=-1;
char stack[MAX];
int main( )
{
char a[MAX],ch;
int op1,op2,value,i;
printf("enter valid postfix expression:\t");
gets(a);
printf("\n the value for the expression is:\t");
for(i=0;a[i]!='\0';i++)
{
ch=a[i];
if(isdigit(ch))
push((double)(ch-'0'));
else
{
op2=pop( );
op1=pop( );
value=oper(ch,op1,op2);
push(value);
}/*end for else*/
}/*end for loop*/
printf("%lf",pop( ));
return 0;
}/*end for main( )*/
```

Input & Output:-

Exp.No. 10	Write a C program that uses functions to perform the following operations on singly linked list. i) Creation ii) Insertion iii) Deletion iv) Traversal
Dt.	

Aim:-Write a C program that uses functions to perform the following operations on singly linked list. i) Creation ii) Insertion iii) Deletion iv) Traversal

Source Code:-

```
#include<stdio.h>
#include<stdlib.h>
Typedef structsll
{
        int data;
        structsll *link;
}node;
node *start,*prev;
node *create(node *);
node *insert(node *);
node *del(node *);
void display(node *);
int main( )
{
        node *start=NULL;
        intch;

        start=create(start);
        display(start);
        do
        {
                printf("\n");
                printf("1. Insert\n"); printf("2. delete\n"); printf("3. Exit\n");
                printf("Enter ur choice:");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1: start=insert(start);
                                        display(start);
                                        break;
                        case 2:start=del(start);
                                        display(start);
                                        break;
                        case 3: exit(0);
                        default: printf("\n u have entered wrong choice");
                }
        }while(ch<=3);
return 0;
}/*end for main*/
```

```
node *create(node *start)
{
	node *temp;
	int item;
	do
	{
		printf("\n enter item and to stop press -999:\t");
		scanf("%d",&item);
		temp=(node *)malloc(sizeof(node));
		temp->data=item;
		temp->link=NULL;
		if(start==NULL)
		start=temp;
		else
		prev->link=temp;
		prev=temp;
	}while(item!=-999);
	return start;
}/*end for create*/

node *insert(node *start)
{
	node *newnode,*temp;
	intpos,item,i;
	printf("\n enter data to insert:");
	scanf("%d",&item);
	printf("\n enter position to insert:");
	scanf("%d",&pos);
	newnode=(node *)malloc(sizeof(node));
	newnode->data=item;
	if((pos==1)||(start==NULL))
	{
		newnode->link=start;
		start=newnode;
	}/*end for if*/
	else
	{
		temp=start;
		i=2;
		while((i<pos)&&(temp->link!=NULL))
		{
			temp=temp->link;
			i++;
		}/*end for while*/

		newnode->link=temp->link;
		temp->link=newnode;
	}/*end for else*/
	return start;
}/*end for insert*/
```

```
node *del(node *start)
{
	node *temp,*prev;
	int item;
	printf("\n enter item to delete:");
	scanf("%d",&item);
	if(start==NULL)
		printf("\n cant delete. list is empty");
	else if(start->data==item)
		start=start->link;
	else
	{
		temp=start;
		while((temp!=NULL)&&(temp->data!=item))
		{
			prev=temp;
			temp=temp->link;
		}
		if(temp==NULL)
			printf("\n element not found");
		else
			prev->link=temp->link;
	}
		return start;
}/*end for del*/

void display(node *start)
{
	printf("\n elements in list are:\n");
	while(start!=NULL && start->data!=-999)
	{
		printf("%d->",start->data);
		start=start->link;
	}

}/*end for display*/
```

Input & Output:-

Exp.No. 11	**DOUBLY LINKED LIST IMPLEMENTATION**
Dt.	

Aim:- Write a C program that uses functions to perform the following operations on Doubly linkedlist. i) Creation ii) Insertion iii) Deletion iv) Traversal

Source Code:-

```
#include <stdio.h>
#include<stdlib.h>
struct dnode
{
struct dnode*prev;
int data;
struct dnode*next;
};
struct dnode *start=NULL;
void insert(int);
void remov(int);
void display( );
int main( )
{
int n,ch;
do
{
   printf("Operations on doubly linked list\n");
   printf("1. Insert \n2.Remove\n3. Display\n0. Exit\n");
   printf("Enter Choice 0-4? : ");
   scanf("%d",&ch);
   switch(ch)
   {
   case 1: printf("Enter number: ");
         scanf("%d",&n);
         insert(n);
         break;
   case 2: printf("Enter number to delete: ");
         scanf("%d",&n);
         remov(n);
         break;
   case 3: display( );
         break;
   }
}while(ch!=0);
}
```

```
void insert(int n)
{
   struct dnode *nptr,*temp=start;
   nptr=malloc(sizeof(struct dnode));
   nptr->data=n;
   nptr->next=NULL;
   nptr->prev=NULL;
   if(start==NULL)
   {
      start=nptr;
   }
   else
   {
      while(temp->next!=NULL)
      temp=temp->next;
      nptr->prev=temp;
      temp->next=nptr;
   }
}

void remov(int n)
{
struct dnode *temp=start;
while(temp!=NULL)
{
   if(temp->data==n)
   {
      if(temp==start)
      {
      start=start->next;
      start->prev=NULL;
      }
      else
      {
         if(temp->next==NULL)
            temp->prev->next=NULL;
         else
         {
         temp->prev->next=temp->next;
         temp->next->prev=temp->prev;
         }
      free(temp);
      }
      return ;
   }//end or if
temp=temp->next;
}//end for while
printf("%d not found.\n",n);
}//end for function
```

```
void display( )
{
   struct dnode*temp=start;
   while(temp!=NULL)
   {
      printf("%d\t",temp->data);
      temp=temp->next;
   }
   printf("\n");
}
```

Input & Output:-

Exp.No. 12	CIRCULAR LINKED LIST
Dt.	

Aim:- Write a program that uses functions to perform the following operations on circular linked list i)Creationii)insertion iii)deletion iv) Traversal

Source Code:-

```
#include<stdio.h>
#include<stdlib.h>
struct Node;
typedef struct Node * PtrToNode;
typedef PtrToNode List;
typedef PtrToNode Position;

struct Node
{
int e;
Position next;
};

void Insert(int x, List l, Position p)
{
Position TmpCell;
TmpCell = (struct Node*) malloc(sizeof(struct Node));
if(TmpCell == NULL)
   printf("Memory out of space\n");
else
{
   TmpCell->e = x;
   TmpCell->next = p->next;
   p->next = TmpCell;
}
}

int isLast(Position p, List l)
{
return (p->next == l);
}

Position FindPrevious(int x, List l)
{
Position p = l;
while(p->next != l && p->next->e != x)
   p = p->next;
return p;
}
```

```
Position Find(int x, List l)
{
Position p = l->next;
while(p != l && p->e != x)
   p = p->next;
return p;
}

void Delete(int x, List l)
{
Position p, TmpCell;
p = FindPrevious(x, l);
if(!isLast(p, l))
{
   TmpCell = p->next;
   p->next = TmpCell->next;
   free(TmpCell);
}
else
   printf("Element does not exist!!!\n");
}

void Display(List l)
{
printf("The list element are :: ");
Position p = l->next;
while(p != l)
{
printf("%d -> ", p->e);
p = p->next;
}
printf("\n");
}

int main( )
{
int x, pos, ch, i;
List l, ll;
l = (struct Node *) malloc(sizeof(struct Node));
l->next = l;
List p = l;
printf("CIRCULAR LINKED LIST IMPLEMENTATION OF LIST ADT\n");
do
{
printf("1. INSERT\t 2.DELETE\t 3.FIND\t 4.PRINT\t 5. QUIT\nEnter the cho ice :: ");
scanf("%d", &ch);
```

```
switch(ch)
{
case 1: printf("Enter the element to be inserted :: ");
      scanf("%d",&x);
      printf("Enter the position of the element :: ");
      scanf("%d",&pos);
      for(i = 1; i < pos; i++)
      {
         p=p->next;
      }
      Insert(x,l,p);
      break;
case 2: p = l;
      printf("Enter the element to be deleted :: ");
      scanf("%d",&x);
      Delete(x,p);
      break;
case 3: p = l;
      printf("Enter the element to be searched :: ");
      scanf("%d",&x);
      p = Find(x,p);
      if(p == l)
         printf("Element does not exist!!!\n");
      else
         printf("Element exist!!!\n");
      break;
case 4:Display(l);
      break;
}// end for switch
}while(ch<5);
return 0;
}//end for while
```

Input & Output:-

Exp.No. 13	**BINARY TREE TRAVERSALS**
Dt.	

Aim:- Write a program to create a binary search tree of integers and perform the following operations
i)insert a node
ii)in-order traversal
pre-order traversal
post-order traversal

Source code:-

```
#include<stdio.h>
#include<stdlib.h>
#include"InsertAndTraversals.c"
int main( )
{
int x, op;
BSTNODE root = NULL;
        while(1)
        {
                printf("1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder
                                Trav ersal 5.Exit\n");
                printf("Enter your option : ");
                scanf("%d", &op);
                switch(op)
                {
                        case 1: printf("Enter an element to be inserted : ");
                                        scanf("%d", &x);
                                        root = insertNodeInBST(root,x);
                                        break;
                        case 2:if(root == NULL)
                                {
                                        printf("Binary Search Tree is empty.\n");
                                }
                                else
                                {
                                                printf("Elements of the BST (in-order
                                                    traversal): ");
                                                inorderInBST(root);
                                                printf("\n");
                                        }
                                        break;
                        case 3:if(root == NULL)
                                        {
                                                printf("Binary Search Tree is empty.\n");
                                        }
                                        else
                                        {
```

```
                                        printf("Elements of the BST (pre-order
                                          traversal): ");
                                        preorderInBST(root);
                                        printf("\n");
                                }
                                break;
                        case 4:if(root == NULL)
                                {
                                printf("Binary Search Tree is empty.\n");
                                }
                                else
                                {
                                        printf("Elements of the BST (post-order
                                                traversal): ");
                                        postorderInBST(root);
                                        printf("\n");
                                }
                                break;
                        case 5: exit(0);
                }
        }
}
```

InsertAndTraversals.c

```
struct node
{
int data;
struct node *left, *right;
};

typedef struct node *BSTNODE;

BSTNODE newNodeInBST(int item)
{
        BSTNODE temp = (BSTNODE)malloc(sizeof(struct node));
        temp->data = item;
        temp->left = temp->right=NULL;
        return temp;
}
```

```
void inorderInBST(BSTNODE root)
{
	if(root!=NULL)
	{
	inorderInBST(root->left);
	printf("%d ",root->data);
	inorderInBST(root->right);
	}
}

void preorderInBST(BSTNODE root)
{
	if(root!=NULL)
	{
	printf("%d ",root->data);
	preorderInBST(root->left);
	preorderInBST(root->right);
	}
}

void postorderInBST(BSTNODE root)
{
	if(root!=NULL)
	{
	postorderInBST(root->left);
	postorderInBST(root->right);
	printf("%d ",root->data);
	}
}

BSTNODE insertNodeInBST(BSTNODE node, int ele)
{
	if(node==NULL)
	{
		printf("Successfully inserted.\n");
		return newNodeInBST(ele);
	}
	if(ele<node->data)
		node->left=insertNodeInBST(node->left,ele);
	else if(ele>node->data)
		node->right=insertNodeInBST(node->right,ele);
	else
		printf("Element already exists in BST.\n");
return node;
}
```

Input & Output:-

Exp.No. 14	LINEAR SEARCH AND BINARY SEARCH
Dt.	

14(a)Linear Search using recursion:-

Source Code:-

```
#include<stdio.h>
void ls(int [ ],int,int);
int main( )
{
int a[50],i,n,key;

printf("\n Enter n value:\n");
scanf("%d",&n);
printf("\n Enter %d elements :\n",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\n Enter key element to be searched:\n");
scanf("%d",&key);
ls(a,n,key);
return 0;
}

void ls(int a[ ],int n,int key)
{
int f=0;
if(a[n-1]==key)
{
printf("\n search is successful.\n");
printf("The element %d is found at position %d ",key,n);
f=1;
}
else
{
if(n==0&&f==0)
printf("\nsearch is unsuccessful.");
else
ls(a,n-1,key);
}
}
```

Input & Output:-

14(b)Linear Search using non-recursion:-

Source Code:-

```
#include<stdio.h>
#include<conio.h>
int ls(int [ ],int,int);
int main( )
{
int a[50],i,n,key,pos;
printf("\n Enter n value:\n");
scanf("%d",&n);
printf("\n Enter %d elements:\n",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\n Enter key element to be searched:\n");
scanf("%d",&key);
pos=ls(a,n,key);
if(pos>=0)
{
printf("\n Search is successful.");
printf("\n The element %d is found at position %d.",key,pos+1);
}
```

```
else
printf("\n search is unsuccessful.");
}
int ls(int a[ ],int n,int key)
{
int i;
for(i=0;i<n;i++)
{
if(a[i]==key)
return i;
}
return -1;
}
```

Input & Output:-

14(c)Binary Search using recursion:-

Source Code:-

```
#include<stdio.h>
#include<conio.h>
int bs(int [ ],int,int,int,int);
int main( )
{
int a[50],i,n,key,low,high,pos;
printf("\n Enter n value:\n");
scanf("%d",&n);
printf("\n Enter %d elements in ascending order:\n",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\n Enter key element to be searched:\n");
scanf("%d",&key);
low=0,high=n-1;
pos=bs(a,n,key,low,high);
if(pos>=0)
{
```

```
printf("\n Search is successful.");
printf("\n The element %d is found at position %d.",key,pos+1);
}
else
printf("\n Search is unsuccessful.");
return 0;
}

int bs(int a[],int n,int key,int low,int high)
{
int mid;
if(low>high)
return -1;
else
{
mid=(low+high)/2;
if(key==a[mid])
return mid;
else if(key>a[mid])
bs(a,n,key,mid+1,high);
else
bs(a,n,key,low,mid-1);
}
}
```

Input & Output:-

14(d)Binary Search using non recursion:-

Source Code:-

```
#include<stdio.h>
#include<conio.h>
int main( )
{
int a[50],n,key,i,pos;
clrscr();
printf("\n Enter n value:\n");
scanf("%d",&n);
printf("\n Enter %d elements in ascending order:\n",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\n Enter key element to be searched:\n");
scanf("%d",&key);
pos=bs(a,n,key);
if(pos>=0)
{
printf("\n Search is successful.");
printf("\n The element %d is found at position %d.",key,pos+1);
}
else
printf("\nSearch is unsuccessful.");
}

bs(int a[ ],int n,int key)
{
int low,mid,high;
low=0,high=n-1;
while(low<=high)
{
mid=(low+high)/2;
if(key==a[mid])
return mid;
else if(key>a[mid])
low=mid+1;
else if(key<a[mid])
high=mid-1;
}
return -1;
}
```

Input & Output:-

Exp.No. 15	**BUBBLE SORT, SELECTION SORT AND INSERTION SORT**
Dt.	

Aim:-Write a C program that implements the following sorting methods to sort a given list of integers in ascending order
i) Bubble sort
ii) Selection sort
iii)Insertion sort

15(a) Bubble Sort

Source Code:-

```
#include<stdio.h>
int main( )
{
int a[50],i,j,n,temp;
printf("\n Enter n value:\n");
scanf("%d",&n);
printf("\n Enter %d elements:\n",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\n Before sorting elements are:\n");
for(i=0;i<n;i++)
printf("%d\t",a[i]);
for(i=0;i<n-1;i++)
{
for(j=i+1;j<=n;j++)
{
if(a[i]>a[j])
{
temp=a[i];
a[i]=a[j];
a[j]=temp;
}/*end for if block*/
}/*end for j loop*/
}/*end for i loop*/
printf("\nAfter sorting elements are:\n");
for(i=0;i<n;i++)
printf("%d\t",a[i]);
}
```

Input & Output:-

15(b) Selection Sort

Source Code:-

```
#include<stdio.h>
int main( )
{
int a[50],i,j,n,small,temp;
printf("\n Enter n value:\n");
scanf("%d",&n);
printf("\n Enter %d elements :\n",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\n Before sorting elements are:\n");
for(i=0;i<n;i++)
printf("%d\t",a[i]);
for(i=0;i<n-1;i++)
{
small=i;
for(j=i+1;j<n;j++)
{
if(a[j]<a[small])
small=j;
}/*end for j loop*/
temp=a[i];
a[i]=a[small];
a[small]=temp;
}/*end for i loop*/
printf("\n After sorting elements are:\n");
for(i=0;i<n;i++)
printf("%d\t",a[i]);
return 0;
}
```

Input & Output:-

15(c) Insertion Sort

Source Code:-

```
#include<stdio.h>
int main()
{
   int a[20],i,n,j,temp;
   printf("How many elements ");
   scanf("%d",&n);
   printf("Enter array elements\n");
   for(i=0;i<n;i++)
   scanf("%d",&a[i]);
   /* Insertion sort */
   for(i=1;i<n;i++)
   {
      temp=a[i];
      j=i-1;
      while(j>=0 && a[j]>temp)
      {
         a[j+1]=a[j];
         j--;
      }
      a[j+1]=temp;
   }
   for(i=0;i<n;i++)
   {
      printf("%d\t",a[i]);
   }
return 0;
}
```

Input & Output:-